In the Claims:

1. (Original) A packet accelerator device, comprising:

a packet header parser configured to parse packet header fields from incoming packets directed toward a host;

a processing mechanism configured to perform packet re-assembly on packets determined to have valid connections with said host;

an address filter configured to identify data parsed from said packet header fields needed for packet re-assembly and place that data in a local memory directly accessible by said processing mechanism.

2. (Original) The packet accelerator according to Claim 1, wherein said processing mechanism is an embedded CPU within the packet accelerator.

3. (Original) The packet accelerator according to Claim 1, wherein said incoming packets are TCP packets.

4. (Original) The packet accelerator according to Claim 1, wherein:

said address filter is further configured to,

identify if the incoming packet is part of an established connection with said host,

reject the incoming packet if it is not part of an established connection, and

forward the incoming packet to said processing mechanism if it is part of an established connection.

5. (Original) The packet accelerator according to Claim 1, wherein said address filter comprises:

a hashing mechanism configured to determine an index based on at least part of the parsed header fields;

a connection table containing connection information indexed by said hashing mechanism; and

a forward engine configured to retrieve connection table values from said connection table corresponding to the incoming packets and compare the retrieved connection table values with the incoming;

wherein said forward engine is further configured to,

discard incoming packets that do not have matching connection information in said connection table; and

forward the incoming packets to the processing mechanism for re-assembly if the incoming packets have matching connection information in said connection table.

6. (Original) The packet accelerator according to Claim 5, wherein said forward engine is further configured to retrieve additional connection table values from said connection table when more than one connection has been stored by reference to said index.

7. (Currently Amended) A ~~The~~ packet accelerator ~~according to Claim 6~~ device, comprising:

a packet header parser configured to parse packet header fields from incoming packets directed toward a host;

a processing mechanism configured to perform packet re-assembly on packets determined to have valid connections with said host; and

an address filter configured to identify data parsed from said packet header fields needed for packet re-assembly and place that data in a local memory directly accessible by said processing mechanism;

wherein:

said address filter comprises,

a hashing mechanism configured to determine an index based on at least part of the parsed header fields,

a connection table containing connection information indexed by said hashing mechanism, and

a forward engine configured to,

retrieve connection table values from said connection table corresponding to the incoming packets and compare the retrieved connection table values with the incoming,

discard incoming packets that do not have matching connection information in said connection table,

forward the incoming packets to the processing mechanism for re-assembly if the incoming packets have matching connection information in said connection table, and

retrieve additional connection table values from said connection table when more than one connection has been stored by reference to said index; and

said connection table comprises a set of first connection address data, each first connection address data is stored at a hashed index location and includes a pointer that is either null, indicating the first connection address data is the only connection address saved at its corresponding hashed index location, or pointing to a next connection address data indicating a next connection address saved at a same hashed index.

8. (Original) A method of accelerating packet re-assembly, comprising the steps of:

parsing a header fields of an incoming packet to determine data needed for packet re-assembly;

forwarding the packet to be re-assembled to a re-assembly mechanism; and

placing the data needed for packet re-assembly in a local memory directly accessible by said re-assembly device.

9. (Original) The method according to Claim 8, further comprising the step of identifying if the incoming packet is part of an established connection with said host.

10. (Original) The method according to Claim 9, wherein said step of determining if the incoming packet is part of an established connection comprises the steps of:

hashing at least part of the parsed header fields to determine an index into a connection table;

retrieving connection information from the connection table based on said index;

comparing the connection information retrieved from the connection table to connection information from the parsed header fields; and

if the connection information from the connection table matches the connection information from the parsed header fields, then, identifying the incoming packet as being part of an established connection with said host.

11. (Original) The method according to Claim 10, wherein said step of hashing comprises performing a polynomial CRC calculation on a predetermined number of lower bits of a TCP connection address of said incoming packet.

12. (Original) The method according to Claim 10, wherein said predetermined number of lower bits comprises a number of bits needed to identify a number of connections supported by said host.

13. (Currently Amended) A ~~The~~ method of accelerating packet re-assembly ~~according to Claim 10~~, ~~further~~ comprising the steps of:

parsing a header fields of an incoming packet to determine data needed for packet re-assembly;

forwarding the packet to be re-assembled to a re-assembly mechanism;

placing the data needed for packet re-assembly in a local memory directly accessible by said re-assembly device; and

identifying if the incoming packet is part of an established connection with said host;

wherein:

said step of determining if the incoming packet is part of an established connection comprises the steps of:

hashing at least part of the parsed header fields to determine an index into a connection table;

retrieving connection information from the connection table based on said index;

comparing the connection information retrieved from the connection table to connection information from the parsed header fields;

if the connection information from the connection table matches the connection information from the parsed header fields, then, identifying the incoming packet as being part of an established connection with said host; and

if the connection information from the connection table does not match the connection information from the parsed header, then,

determining if any additional connection addresses are hashed into the connection table at said index, and

if additional connection address are hashed into the connection table at said index, then,

retrieving the additional connection addresses,

comparing the additional connection addresses to the connection information from the parsed header fields, and

if any of the additional connection addresses match the parsed header fields, identifying the incoming packet as being part of an established connection with said host.

14. (Original) The method according to Claim 13, wherein:

said step of determining if any additional connection addresses are hashed into the connection table at said index comprises determining if a next pointer field at said index of the connection table is null, indicating no other connection addresses are hashed into the index, or, not null, indicating that additional connection addresses have been hashed into the connection table at the index; and

said step of retrieving the additional connection addresses comprises reading data pointed to by the next pointer field in the connection table at the index and each subsequent next pointer field of the read data.

15. (Cancel)

16. (Cancel)

17. (Original) A packet accelerator device, comprising:

a packet header parser configured to parse packet header fields from incoming packets directed toward a host;

a processing mechanism configured to perform packet re-assembly on packets determined to have valid connections with said host; and

an address filter configured to identify data parsed from said packet header fields needed for packet re-assembly and place that data in a local memory directly accessible by said processing mechanism;

wherein:

said processing mechanism is an embedded CPU within the packet accelerator;

said incoming packets are TCP packets;

said address filter is further configured to,

identify if the incoming packet is part of an established connection with said host,

reject the incoming packet if it is not part of an established connection, and

forward the incoming packet to said processing mechanism if it is part of an established connection; and

said address filter comprises,

a hashing mechanism configured to determine an index based on at least part of the parsed header fields,

a connection table containing connection information indexed by said hashing mechanism, and

a forward engine configured to,

retrieve connection table values from said connection table corresponding to the incoming packets and compare the retrieved connection table values with the incoming,

discard incoming packets that do not have matching connection information in said connection table, and

forward the incoming packets to the processing mechanism for re-assembly if the incoming packets have matching connection information in said connection table.

18. (Original) A device for packet re-assembly, comprising:

means for parsing headers of incoming packets directed toward a host;

means for constructing a frame status;

means for performing packet re-assembly on the incoming packet using said frame status; and

means for storing said frame status in a memory local to said means for constructing;

wherein:

said frame status includes information needed to perform packet re-assembly; and

said means for constructing performs said packet re-assembly by using said frame status and not having to access a packet buffer for information contained in said packet header.

19. (Original) The device according to Claim 18, wherein said means for performing packet re-assembly comprises means for performing TCP packet re-assembly on the incoming packet.

20. (Original) The device according to Claim 18, wherein said frame status comprises:

a frame pointer configured to point to a memory location with the packet buffer of the incoming frame;

at least one index/pointer field capable of being used to determine a connection address corresponding to said incoming packet;

segment data comprising information that identifies whether a full segment of packets corresponding to the incoming packet have been received;

sequence data identifying an order of the incoming frame within the segment corresponding to the incoming packet;

length of the incoming packet; and

an offset identifying a starting position of a payload of the incoming packet.

21. (Original) The device according to Claim 18, further comprising:

means for identifying established connections with said host;

means for comparing incoming packets with said established connections to determine if the incoming packets are part of an established connection with said host; and

means for discarding the incoming packets if they are not part of established connections with said host.

22. (New) The method according to Claim 18, wherein:

said address filter identifies if the incoming packet is part of an established connection by,

performing a polynomial CRC calculation on a predetermined number of lower bits of a TCP connection address of said incoming packet,

indexing into a connection table with the polynomial CRC calculation as an index, the connection table comprising a first valid connection for the index and a list of additional valid connections for the same index, and

comparing the TCP connection to each of the first valid connection and each additional valid connection until a match is made.

23. (New) The method according to Claim 22, wherein the connection table references two separate tables, the first table providing the TCP connection address for a primary connection and the second table comprising all secondary TCP connections hashed into a same index.